Distributing Information for Collaborative Filtering on Usenet Net News

by

David A. Maltz

Submitted to the
Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degrees of

Bachelor of Science in Computer Science and Engineering
and
Master of Science in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 1994

Distributing Information for Collaborative Filtering on Usenet Net News

by

David A. Maltz

Submitted to the Department of Electrical Engineering and Computer Science
on May 12, 1994, in partial fulfillment of the
requirements for the degrees of
Bachelor of Science in Computer Science and Engineering
and
Master of Science in Electrical Engineering and Computer Science

Abstract

As part of the "Information Revolution," the amount of raw information available to computer users has increased as never before. Unfortunately, there has been a corresponding jump in the amount of unrelated information users must search through in order find information of interest. Harnessing the power of multiple users to form a collaborative filter provides a robust way of helping direct users to the information that will be most useful to them. To test this idea, we have designed a large scale collaborative filtering system tuned to help users extract information from Usenet Net News.

In this thesis we demonstrate a system for collaborative filtering that can scale up to encompass the large distributed information sources of which Usenet Net News is an example. Our system provides varying levels of anonymity to protect the interests of the users, as well as means of minimizing the load placed on the existing information source. We believe the system will be especially good at three tasks: supporting users as they explore new areas of interest; providing users a way of keeping up to date on areas they already have familiarity with; and at providing extra information to other filters.

Thesis Supervisor: Karen Sollins
Title: Research Scientist, Laboratory for Computer Science

Thesis Supervisor: David Goldberg
Title: Member of the Research Staff, Xerox Palo Alto Research Center

Acknowledgments

The preparation of this thesis has benefited from the insights of many people, but none more than my two advisors: David Goldberg and Karen Sollins. Their patience in providing guidance and feedback has been phenomenal, and their skill at pointing me towards relevant references and resources has been acute and unerring.

I would also like to thank the many people at Xerox's Palo Alto Research Center. Over my three stays there they provided a fertile and supportive environment in which to learn how to do research. Special thanks go to Peter Pirolli who helped perform the initial analysis of data from the user behavior study, and to Ron Frederick and Berry Kercheval who provided humor and UNIX wizardry when they were frequently needed.

Finally, I would like to dedicate this work to my friends and family. I stand in admiration of my parents and the wisdom, compassion, and strength they have shown in working through the trials of life. I would not have made it this far without the love and support they have given me over the years. And to Sasha, who has in many ways made my undergraduate years worth doing.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Connectivity, networking, and the National Information Infrastructure have become the buzzwords of the day. Underlying the excitement of these times is the promise that each of us will soon have unlimited access to a computer which will cheaply bring to us information from sources around the world. We will be in direct contact with all the world's repositories of information – no matter how small or large – and in direct contact with the experts and people who create those repositories. Just like the upswell of creation and learning which followed the development of the printing press and the widespread access to information it created, we anticipate a new surge of knowledge that will enrich our lives.

Unfortunately, our situation parallels that of the printing press in more ways than one. As the first libraries were built and books became available to larger groups of people, a new problem arose. Once buildings could be filled with more piles of books than any human could possibly read in a lifetime, how could people *find* books on the topics they wanted? The solution to that problem evolved into an entire field called Library Science. A solution to the problem of finding useful information on a global network promises to be no easier.

In this thesis we describe one possible system for helping users find the information they want from the stacks of the global internet. The system works by making it possible for users at distant sites to take advantage of each others' experiences with information sources. These experiences can then be called on by users to direct their information searches. Our approach is derived from the Information Tapestry project at Xerox PARC and we use their name to describe it - collaborative filtering.

In the sections that follow we first present an example of the type of information system we will be working with, and then describe the various types of filtering currently in use for that information system. In chapter 2 we form a specification of what we want our filtering system to accomplish. For chapter 3 we discuss the constraints imposed on our system by the information system we are filtering. In chapter 4 we develop the design of our system, and we conclude by describing user's opinions and usage of our system.

1.1 A glimpse of the future

The array of new information sources available over the networks seems bewildering at first. A brief glance through the literature surrounding the National Information Infrastructure shows examples of many different types of commercial products. Examples of such products are: Companies who specialize in creating "custom storefront" services for other companies so users can purchase items over the net; Software companies setting up product information services to field questions, updates and product enhancements over the internet network rather than the phone network; and quote services that provide constantly updated stock market quotes and information for investors.

There has also been growth in the non-commercial sector. The Library of Congress has begun to make its special exhibits and catalogs available through online services. The World Wide Web system allows anyone with a machine on the Internet to create hypertext multimedia documents and link them to other documents at other sites. [7] The Internet Gopher system provides a consistent interface to many different types of information sources ranging from full-text retrieval services to school catalogs and course schedules.[1]

Characteristic of many of these new information systems are the following properties:

- **Article based:** Information is added to them in small, discrete chunks which are reasonably self contained.

- **Independently created:** Information chunks are added to the system by large numbers of users resident at different sites or at different times.

- **Multiply accessed:** Each chunk of information can expect to be read or accessed by a large number of people over the course of its life.

Another example of an information system with these properties is Usenet Net News. The Usenet is a loosely organized network of heterogeneous computers stretching across all 7 continents. These computers communicate via a number of protocols such as **uucp**[19] and TCP/IP data streams. A primary use of the Usenet network is the exchange of Usenet Net News.[28] Net News functions like a huge distributed bulletin board system such that messages created at one site are eventually seen at all sites on Usenet.

Information on Net News takes the form of *articles* written by individual users. Users enter articles into the Net News system by *posting* them. Each site which participates in Usenet Net News runs a program called a *news server* that exchanges articles with news servers at other sites using the Network News Transfer Protocol (NNTP).[12] The news server stores all the articles it receives for a number of days and deletes older articles to make room for the new. Until the articles are deleted, the news server makes them available to be read by users at the site. Users request articles from the news server using any of a number of *news reader* programs which provide a user interface for reading and browsing the articles.

To help users find articles they are interested in reading, the articles are arranged into a hierarchy of *newsgroups* which is organized by topic. To keep discussions on topic, some newsgroups are *moderated*. Any article posted to a moderated newsgroups is first sent to a human moderator who decides whether or not the article will be distributed across the Usenet.

Usenet Net News is currently growing to an enormous size. Estimates show that there are over 2.6 million users of Net News at some 87 thousand sites throughout the world. These users generate over 26 thousand new articles a day, amounting to 57 Mbytes of data.[22] In the past, users attempted to control the number of articles they had to read a day by only subscribing to newsgroups on topics in which they are interested. However, the continuing increase in the number of newsgroups and the number of articles posted daily has had the result that many users are presented with far more messages a day than they can possibly read. To cope with this overload of incoming data, users have adopted reading strategies for dealing with the flood, but these strategies often sacrifice any chance the user has of finding useful information in the data. In fact, one of the few things the users of Net News seem to agree on is that they need a simple way of filtering the available messages to find ones in which they will be interested.[11]

The idea of filtering is not a new one, and many filtering systems of different types have been created in the past, but none of them have all the characteristics that are needed for filtering information systems like Net News. Furthermore, few existing filtering systems take advantage of a key property of networked information systems — namely that many people read each message. Making use of the multiply accessed property to create an effective filtering system is the primary goal of this thesis.

As an overview, the system we developed for filtering Net News enables users to cast votes on the articles they read, and users can either associate their names with these votes or not as they chose. The votes are distributed by a series of methods from where they are cast to where later users might access them. When users go to read the articles available in a newsgroup, they can access the votes cast by previous users and thereby find out whether previous users found those articles to be useful. Before describing our system for filtering Net News in more detail, let us first examine several general strategies that are currently used for filtering.

1.2 Systems for filtering

Most present filtering systems are examples of what Malone has termed a cognitive filter.[18] These filters attempt to select articles for a user to view by determining the value of the information contained in the article to the user. These systems are primarily designed for use by a single user. All pieces of information that might be displayed to the user are first sent through a filter that applies a set of criteria to each piece of information. Articles which meet a sufficient number of the filter's criteria are passed on to the user with the remaining articles being discarded.

Cognitive filters can be roughly broken down into the two classes of user-profile based or rule based filters. The difference between the two lies in how the criteria used for filtering are created. In a user-profile based system, the filtering criteria are usually generated automatically in response to user actions. Many different methods have been developed for storing the filter criteria used in a user-profile system. In contrast, users explicitly enter the rules which are used to filter information in a rule based filter system. Such rules often take the form of boolean or structured query language (SQL) expressions. We will now examine some examples of each type of filtering system.

User-profile based systems attempt to extract patterns from the observed behavior of the user, and then predict which other items from the information stream will be selected or rejected based on the patterns. An example of such systems is the LyricTime project by Shoshana Loeb which compiles a profile of musical tastes for each user of an on-line jukebox.[16] LyricTime uses the user's profile to filter songs from a collection of available music. As the user accepts and rejects songs suggested by the system, the system refines the user's profile with the goal of delivering only songs which the user will approve of. LyricTime stores a user's profile as a series of keywords to look for and a confidence rating to describe how sure the system is that the user approves of songs with the keyword.

Another user-profile based filtering system is the Evolving Agent by Beerud Sheth and Pattie Maes.[26] This system attempts to learn how to filter Net News for a user by creating an new agent to filter for each topic that interests the user. These agents store information about the user's interest profile as a genotype consisting of keywords to look for and associated numerical weights. Articles are filtered by assigning each article a score proportional to the number of keywords it contains scaled by the weights of those keywords. Only articles with a suitably high score are shown to the user. The system improves its ability to filter Net News by using a genetic algorithm to breed the agents and create a new crop of agents better adapted to filtering for the user. Relevance feedback from the user is used as the fitness function which determines which agents survive and reproduce.

INFOSCOPE is an interesting hybrid news reader by Gerhard Fischer and Curt Stevens.[9] Rather than seeing itself as a sieve style filter, INFOSCOPE frames the filtering problem as one of restructuring the information space on a user by user basis to place all the articles relevant to a user in a few accessible locations. INFOSCOPE performs this restructuring by binning new articles into baskets set up by the user to contain the articles relevant to the user's interests. These baskets, called "virtual newsgroups," can draw articles from many Usenet newsgroups, selecting which articles to include in the virtual newsgroup by using either rule or profile based sieve filtering techniques.

1.2.1 Drawbacks of current filtering systems

Systems such as the Evolving Agent, Lyrictime, and INFOSCOPE all suffer from a "cold-start" problem in that new users start off with nothing in their profile and must train a profile from scratch. During the training period the system can't effectively filter for the user, and this initial hump may convince many users to stop using the filtering system, or even give up on the information source entirely. A better system would allow new users some type of access to the experiences of current users to help create an initial profile.

A more general problem with systems that rely on user profiles is that the user can become circumscribed by their profile. The profile only selects articles similar to the ones the user has already read, and new areas which might be of interest can be missed completely. Further, if the user tries to explore areas outside the knowledge domain of the profile, the profile can provide little if any filtering ability, so the user will again confront the cold-start problem and be left facing a staggering amount of data to search through manually. To support the exploration of new areas what is needed is a method for tapping the knowledge of users currently well versed in those areas.

1.2.2 Collaboration as a method for filtering

Collaborative filtering systems can support exploratory searching by providing users with information derived from the experiences of previous users. Collaborative filtering is based on the observation that people are good editors and filters for their friends. Currently, when a user reads an article that he thinks will be of interest to a friend, he copies the article and sends it to the friend. Often the friend will not even know that the information contained in the article is available, and so could not have constructed a filter rule or query to extract the information directly. As suggested by Malone, collaborative filtering takes advantage of the social interactions between people to create a filter.

Another advantage which collaborative filtering systems have over other automatic filtering systems is that we can expect human beings to be better at evaluating documents than a computed function. Automatic filtering systems attempt to find articles of interest to their user, often using some scoring function to evaluate features of the documents and returning the documents with the highest scores. People can effortlessly evaluate features of a document that are important to other people, but would be "AI-complete" to detect automatically. Examples of such features are the writing style and "readability" of a document, or the clarity and forcefulness of an argument the document contains. Imagine the difficulty an automatic filtering system would have figuring out which of two cake recipes is "easier to follow."

A further motivation for collaborative filtering comes from the following observation made by Hill et al.[14] The objects we use in everyday life accumulate wear and tear as a normal part of their use: pages in books become wrinkled, bindings creased, and margins smudged with fingerprints. This wear is usually frowned upon as indicating that the object is wearing out and needs to be replaced. Consider however, how helpful are these markings gathered unobtrusively over time: Reference books will open to the most commonly used pages when dropped on a desk. The well thumbed paperback books in a library are the most commonly read ones. The most frequently used recipes in a cook book will have the most stains. The wear on these objects acts as an index to information they contain.

Now compare the rich environment of real objects to the much poorer one in which computer users operate. When a user reads a computer file he usually has no way of telling whether he is the first person to ever read it, or if he is looking at the most commonly used reference on the system. Collaborative filtering works by associating with computer documents the history of their use. Access to this history provides users with the type of subtle hints that we already take advantage of when making read/don't read decisions in the real world.

1.3 Systems for collaborative filtering

The term collaborative filtering itself was coined by Doug Terry at Xerox PARC as part of the development of the Information Tapestry system for retrieving documents from a growing corpus.[13] Among its other features, Tapestry was the first system to support collaborative filtering in that it allows its users to annotate the documents they read. Other Tapestry users can then retrieve documents to read based not only on the content of the documents themselves, but also on what other users have said about them. Tapestry provides free text annotations as well as explicit "likeit" and "hateit" annotations so users can easily indicate which of the documents they read they found most (or least) valuable.

"A tour through Tapestry" demonstrates how the collaborative filtering abilities of Tapestry can help users process incoming documents with much greater ease than current systems.[27] In part by using annotations placed on documents by his co-workers, the protagonist in "A Tour through Tapestry" is able to focus his attention on the documents most likely to be of interest to him. He harnesses his co-workers' expertise to help him find useful information and he repays this service by providing annotations on the documents he reads. In a situation where each co-worker has expertise in a slightly different area, the information sharing created by document annotations gives everyone in the group access to the area expert without overloading the expert's time.

In its current incarnation, Tapestry suffers from two distinct problems. The first problem is the size of its user base. Because Tapestry is based on a commercial database system it can not be given away freely. Further, Tapestry was not designed for use by large numbers of people at distributed sites. Both these factors combine to limit the pool of potential Tapestry users to researchers at Xerox PARC. Based on anecdotal evidence, this pool does not seem large enough to support a critical mass of users. The vast majority of documents go unannotated, so there is little collaborative information to use when filtering. The second problem with Tapestry is the means by which users enter filters into Tapestry. One common interface to Tapestry requires users to specify requests for information in the form of queries in an SQL-like language. Writing such a query requires the user to have a firm sense of what types of articles he wants to read, which is a hindrance to exploration of new areas. Our goal in this thesis is to describe a collaborative filtering system for Net News that can scale up to handle at least a critical mass of users, and provides those users with a simple method for collaboratively filtering articles.

Collaborative filtering is rapidly gaining popularity as a research topic, and many groups are currently working to develop new strategies for collaborative filtering. Simon is collaborative system being developed by Mark Johnson for use with the World Wide Web system.[10] Users of Simon create "hotlists" which are lists of the interesting World Wide Web pages that they have found. Individual users can use these lists, called "subject spaces," to keep track of their own explorations, but they can also send their subject spaces to a group Simon server. The Simon server will then combine the individual subject spaces to form global maps which can be searched or browsed by Web users at large.

GroupLens by Paul Resnick et al. is a filtering system that combines collaboration with user-profiles. In GroupLens, communities of users rank the articles they read on a numerical scale. The GroupLens system then finds correlations between the ratings users have given the articles. Essentially, a user's profile consists of the ratings that she has given to the articles she has read. When user Jane wishes to filter new articles of information, the ratings other users have given those new articles are combined to form a recommendation for Jane on how interesting the new articles will be for her. The ratings from other users are combined by weighting each user's rating in proportion to how well his user-profile correlates with Jane's. The goal of the system is to identify a peer group of users whose interests are similar to Jane's, and then to use their opinions of new articles to predict whether Jane will like the articles. A key difference between our work and GroupLens is the effort we make to support "exploratory users" who have not yet developed a user profile.

Where are we? In this chapter we presented a brief description of the Usenet Net News system and several of that system's characteristics. We described how most current filtering systems are based around the idea of queries or user-profiles and explained how those techniques do not adequately support all users. We then described the concept of collaborative filtering along with the Tapestry system which first embodied it. In the next chapter, we will examine how the characteristics of Net News make it well suited for collaborative filtering and set out the types of functionality we believe a collaborative filtering system for Net News should have.

Chapter 2

Problem Statement

The key failing we see in conventional filtering strategies is that they do not take advantage of the large number of users who are each performing their own filtering and evaluation of the available articles. Because conventional strategies focus only on each user's private history, these filters are unable to provide effective filtering in any domain that a user has not already experienced. While we can reasonably expect a user to have significant experience in the several areas in which he or she has a special long term interest, a user will probably not have any history available for filtering either articles in a new area of interest, or news items that by their very nature change topic frequently. "If it isn't new, it isn't news," goes the reporter's adage.

2.1 Desired system functionality

We do not believe that collaborative filtering alone will meet the average user's filtering needs. However, we assert that there is a real place to be filled by collaborative filtering, and that collaborative filtering and other filtering methods can be effectively used in tandem to create good filters. The Tapestry system already demonstrates one way to integrate filtering methods by using scoring rules. To demonstrate more uses for collaborative filtering, let us consider some scenarios and describe what queries would be useful to a user in each situation. We will then use these queries as a specification for the types of queries our collaborative filtering system should support.

First, consider a user who has seen a picture of a rock climber and developed a passing interest in climbing. This user might find the newsgroup rec.climbing and want to see whether there is any discussion of interest to her in this newsgroup. She would like to see a list of the articles that members of the group thought were most relevant to the topic of the newsgroup. If these articles look interesting, she may decide that interesting things are discussed in rec.climbing and read further. If the "best" articles in rec.climbing do not interest her, she can go on to investigate rec.skydiving. This is an example of an exploratory user.

Second, consider a user who is interested in Microsoft's patent battles. He has used a keyword or profile based filtering system to create a filter which looks for the words "Microsoft" and "patent" in articles, but due to the large amount of discussion on this topic he still receives too many articles — many of which are redundant, inaccurate, or flames. This user might use a collaborative filter to disregard any article that did not have some number of positive reviews. By cutting off the bottom of the distribution, the user improves the quality of the articles passing on to his keyword filter. This is an example of a refining user.

Third, consider a user who reads comp.arch and wants to be kept up to date on any new developments in the newsgroup but does not want to read all the articles. This user might know some other person who does read all of comp.arch and whose opinion he trusts. Our user would like to see all of the articles the dedicated reader liked; basically asking to see how the group would look if this person were moderating it. This is an example of a custom moderation user.

Fourth, consider a group of people working together on a project and using Net News as a source of information. These people would like to make sure that all of them see any article that any individual thinks is relevant. They do not want to have to each print out the articles and distribute them at meetings since there are likely to be many duplicates. Each of these users would like to be able to combine the opinions of the others and view any article that any user found important. The users basically want to see the articles as if they were being moderated by a collective of the users.

This is an example of group moderation. Custom moderation is really just a subset of group moderation where the group size is one.

2.2 System constraints

The goal of this thesis is to demonstrate a collaborative filtering system capable of supporting exploratory, refining, and group moderation on large scale distributed information systems. Because such systems involve replication or non-local servers, the filtering system design must balance such issues as replication versus response time. Because of the large numbers of articles and users on such systems, data completeness must be weighed against storage requirements. Because of the variety of ways in which data can be grouped the rigidity of data organization must be balanced against the ease of providing data to answer queries.

Usenet Net News is the natural target for a collaborative filtering system implementation for several reasons. First, it is an existing system with over 2.5 million potential users world-wide. Second, it meets all three criteria for an information system on which we expect collaborative filtering to work well — information is manipulated as articles, the articles are reasonably self contained, and each article is read by many people. Third, almost all its vocal users say it suffers from a low signal to noise ratio and is in dire need of more filtering capabilities.

The drawback for trying to implement a collaborative filtering system for a preexisting information system is that our system must fit within the constraints imposed by Usenet Net News. As an example, Usenet users bring with them expectations of how Net News should work with which we must not conflict. Further, Usenet system administrators are more open to some types of software changes than others.

The following list sums up the key engineering requirements and constraints placed on our system by either collaborative filtering in general, or the Usenet Net News domain in particular. Each of these points will be addressed in more detail in the following chapters.

- **Accessability:** Users must be able to control who has access to their opinions on articles of information. Some users may only be willing to contribute information anonymously, while others will want to put their names on their opinions.

- **Low Overhead:** The resources required to transport and store filtering information should be a small fraction of the resources required by the information stream as a whole. This is difficult as we hope to have more people providing filtering information than the number who provide the base information stream.

- **Minimum Hassle:** Since our goal was to create a system that people would actually incorporate into their existing Net News system, our software had to be designed for ease of integration. This is difficult given the large number of platforms on which Net News runs and the many configurations in which Net News systems come.

- **Respect for Conventions:** There are many vocal members of the Net News community with strong opinions on what acceptable social conventions are. Our system should be consistent with these.

- **Streamlined:** The average user spends so little time reading most articles that any operation we expect a majority of users to perform must be exceedingly quick and consistent with the flow of the interface they already use. Further, the collaborative filtering system must respond very quickly to requests for information.

2.3 Summary of the solution

Our basic solution to these constraints is a system of replicated vote servers which store information about the popularity of each article separately from the articles of information themselves. Using their normal news reading clients, users contribute votes for or against articles they process. These votes are sent to the nearest vote server where they are grouped together and shared with other vote servers to create a net-wide collective opinion on the relevance of each article. These aggregate opinions are then used by the news reader clients to filter the articles shown to the user. Users can also make their votes directly available to other users, which allows the custom or group moderation of articles.

Where are we? In this chapter we set out the basic functionality we feel a collaborative filtering system should have and the high level constraints that the system must be designed within. In the next chapter we will discuss in more detail the trade-offs considered in the design of our system and data we gathered to support those decisions. The following chapters then describe the design of our system which was based on these conclusions.

Chapter 3

Issues in Design

In this chapter we will examine the four major issues that influenced our design of the collaborative filtering system. We start by examining the issue of accessibility and who has access to a user's votes. We move on to the key issue of distributing vote information in a fashion that imposes minimum overhead on the Net News system. We continue by noting social conventions of Usenet that constrain our design. Finally, we end by searching for patterns of user behavior that quantify the troubles users have searching for interesting articles.

3.1 Providing levels of accessibility

A primary concern of previous researchers in areas involving collaborative work has been providing users privacy and control over how the data collected from them is used by others.[14][5] "A significant segment of the population wants to protect the privacy of any information related to its demographics, and its interests."[17]

Specific to this project, we assume that there is a large class of users who would be willing to vote for or against articles, but for a multitude of reasons might not want others to find out how they have voted. For these users the collaborative system should provide a level of anonymity at least as good as the current Net News system. Net News by default maintains extensive logs of which articles have been requested by which machines — these logs being useful for managing the system. Someone with sufficient access to either these logs or users' accounts could piece together an accurate picture of what newsgroups and articles a user reads. For our system to provide a level of anonymity akin to Net News means that with sufficient sifting of the system logs it would be possible to find out which users read and voted on which articles, but the voting information is not easily available to anyone but a dedicated person with access to the logs.

Collaborative information contributed anonymously is useful for creating net wide summaries of the interest in particular articles. It is these summaries that will be available to help guide exploratory users. We assume, however, that vote information will be much more useful for collaborative filtering if the information is labeled with the name of the person it came from. Further, this type of identified information will have to be available in order to provide custom moderation of newsgroups.

In our design we provide users with methods of casting votes anonymously or of casting identifiable votes. The users also have indirect control over who can access their identified votes. By supporting the extreme positions, we believe we provide everyone a means of participation with which that they will feel comfortable.

3.2 Controlling the costs of collaborative filtering

In order to implement a collaborative filter, there must be some way of distributing a user's votes for or against articles to the users who might want this information. To be a practical system however, this information exchange must place only a small overhead on the existing information stream. To illustrate that this is nontrivial, let us consider a simple example of how filtering information might be transported.

Consider an average sized news group such as comp.arch. This group is read by about 80,000 people and has an average of 1,000 posts a month so each reader is presented with approximately 30 new messages a day. Let us assume that 1% of the readership (800 people) participate in our filtering system. The simplest way for these users to distribute their votes would be to post their votes as an article to some shadow newsgroup, say comp.arch.votes. The existing Net News system would then automatically distribute the votes to all sites throughout the world, and anyone who wanted to use the votes for filtering could simply have their software read the votes out of the shadow group.

Now consider what is happening to that shadow group. The shadow group is receiving 800 messages *a day*. Further, since we will want to store the votes for at least as long as we store the messages to which they refer and the typical expiration time on Net News is 14 days, the shadow group will have on the order of 11,000 messages in it. To make matters worse, since each message will only contain one person's opinion of a message, our software will have to read all 11,000 messages before it can accumulate the collaborative filtering information we need to filter the 30 new articles of actual information.

There is one benefit to the above system, however, which is that the end user has access to complete information about the source of each vote. Having access to this information allows the end user to be very sophisticated in the design of his filter as the filter can place more weight on some opinions, less on others. The filter could track the long term behavior of opinion providers to determine which are the best predictors or use a clustering algorithm to identify a group of peers with similar interests. This is the approach taken by GroupLens.[23]

There is an unavoidable natural tension between the completeness of the collaborative information we make available for use and the cost we must pay to transport it around. In this case, the completeness of the information refers to our ability to associate an opinion with the human being who created that opinion. Considering that we have already decided to provide anonymous voting the resolution of this tension seems clear: information about the net-wide opinion of an article will be available only in summary form.

Even if there were a way to cheaply transport information about the source of each vote, the collaborative system would still have to find ways of insuring the privacy of its users. By aggregating vote information into summaries we blur the information about where a vote originated from in the same way the U.S. Census Office provides privacy to U.S. citizens by only reporting census information aggregated over cityblock sized units. In terms of minimizing the cost for transporting vote information, summaries are also ideal as information about each article only needs to be stored once. This makes the summary very compact. Further, it is trivial to combine two summaries to form a third which takes up little more space than either of the originals, yet carries the combined information.

There must be some way of accessing non-summarized information, however, if we are to implement group and custom moderation. To provide moderation, we do not need to know the opinion of the net as a whole, but only the opinion of the several people whose judgments we trust. Our system provides a direct, point to point, means of obtaining this information when it is specifically required. This client-pull approach will save network bandwidth in what we assume is the common case of users mainly requesting the opinions of other users at the same or nearby sites. If one set of opinions were to become requested by many nonlocal users (eg: someone started to sell their votes as a moderation service) then other technologies termed Uniform Resource Names (URNS) related to URLs could be used to replicate and distribute the collaborative information.[8]

3.3 Respecting social conventions of the Usenet

Usenet Net News is more than just a distributed bulletin board system. It is a community which, while not having formal laws or rules, has a large body of accepted social conventions that govern the behavior of its participants. Because there are no written rules, what the acceptable conventions are is a topic frequently in dispute. For the purposes of this thesis, we have adopted the following conservative guidelines based on observations of discussions on the newsgroups news.future and news.admin.policy.

- **No Censorship** No articles may be suppressed. If a user desires to see all the articles posted to a group, the system must provide them. There must always be an obvious way of shutting the filtering off.

- **No Alterations** The system may not alter in any way an existing article — to do so violates the rights of the article's author. In particular, it would not be permissible to add extra header fields to an article which were used to indicate the net-wide opinion of an article.[1]

- **Decentralized** Avoid centralized servers or clearinghouses. Usenet is an anarchy with very little central control. There is no one site or entity with responsibility for all of Usenet, and no one site or entity is able to control, censor, or shut-down Usenet. This decentralization is useful from a legal standpoint as there is no organizational entity to sue, and important to administrators who need and want control over their sites. Furthermore, much of the robustness of Net News stems from its decentralized nature.

It was primarily to avoid making alterations to existing articles that we decided to store the vote information separately from the articles themselves. In the absence of this restriction, it would be convenient to store the votes received by an article as extra header fields in the article.[2]

3.4 Hooking in the collaborative filter

If any collaborative filtering system for Usenet Net News is ever to be successful, it will need to meet the requirements of not only the users, but also the Net News system administrators who will have to build, install and maintain the system. These people are often volunteers or have been volunteered for the job, so our system must place a minimal load on them.

Net News systems consist of two primary parts: client news readers which are short lived programs started by individual users, and news server software which

[1] An example of the furor raised over this issue can be found in the archives of the news.* groups under the keywords "Richard Depew," a proponent of a system called "retro-moderation."

[2] A technical drawback to storing vote annotations on the articles themselves is that doing so would increase the disk load on the server machine. The arrival of new votes at a server would cause random disk accesses to update article header fields. These extra disk access would interfere with user requests to retrieve articles, causing disk contention to increase.

is constantly running. Many designs for a collaborative filtering system could be radically simplified if they had support from the news server software. Unfortunately, to be consistent with our goal of causing minimum hassle we have chosen to make changes only to the client news readers. There are three reasons for this.

First, informal opinions show that most news administrators have spent large quantities of time and chewing gum sticking together and customizing their news servers for their particular sites. The presence of these many code changes make the distribution of usable patches difficult and makes many administrators wary of making any further changes. News readers on the other hand are frequently compiled and built directly from distribution files with the only customizations being made in configuration files.

Second, news server processes are long lived and maintain large amounts of state. If the collaborative software were to cause a bug in a news server, large quantities of data might be irrecoverably lost. News readers on the other hand, are relatively state free. If the collaborative software caused a news reader to malfunction, it would be easy to back down and begin using an older version again.

Third, it was not our intention to carefully study the user interface issues of exactly how to present collaborative filtering information to users, or to how use the information to best filter and display articles. By making tools that help designers create news readers which support collaborative filtering, we aim to help others explore these human-computer interface issues.

We believe that placing the changes needed to support collaborative filtering in the client news reader programs will both minimize the hassle placed on system administrators and provide a more open system.

3.5 Behaviors of Net News Users

Given the intention of creating a system that will be useful for filtering Net News, it is crucial that the system be designed in such a way that the system supports the users. In order to help our users, we needed to find out how they currently go about reading Net News so our system could be consistent with their demands. To achieve this end, we instrumented two of the news reader programs in use at PARC to record information about how their users read Net News. The subsections that follow describe our method for collecting the experimental data and a summary of our analysis.

3.5.1 Obtaining data on user behavior

We chose to study the users of the **nn** and **xrn** news readers because **nn** and **xrn** are two popular news readers in wide spread usage with very different user interfaces. The interface to the **nn** news reader is text based, while **xrn** has a graphical user interface. The benefit of studying two news readers with different interfaces is that differences in reading behavior caused by the user interfaces could be isolated. Further, when we subsequently modified the news readers as will be described in later chapters, we had a larger pool of potential users from which to draw.

To obtain data which would give an accurate picture of the behavior of most users, we wanted to collect data from all the **nn** and **xrn** users at PARC. We chose not to ask for a group of volunteers to let us collect data on their reading habits for fear of biasing our survey with a self-selected sample group. Unfortunately, this choice meant that since we did not have explicit permission from each individual to record information about him or her, we had to record data in such a way that no individual could be identified from collected data.

To preserve the anonymity of the user community all the data was recorded in a global append-only log file. The log file consists of a series of records; each record representing one *session* with a news reader — a session being defined as the time between when a user starts a news reader program to when he exits it or has a four hour period of inactivity. For every newsgroup the user looked at during the session an entry was made in the log file. We considered an article to have been "read" if the text of the article was displayed to the user. An excerpt of raw data from the global log file is shown in figure 3-1. Because **nn** and **xrn** have different user interfaces, they recorded similar, but not identical, types of information for each newsgroup.

```
-- xrn entry -- Mon Jul 19 15:49:06 1993
  GRP: AVAIL: 1   READ: 1   TIME: 125 *
  GRP: AVAIL: 2   READ: 1   TIME: 1 *
  GRP: AVAIL: 11   READ: 3   TIME: 69 *
  GRP: AVAIL: 7   READ: 3   TIME: 89 *
-- nn entry -- Mon Jul 19 18:16:21 1993
  GRP: AVAIL: 33   READ: 0   TIME_READ: 0   TIME_SCAN: 4665
  GRP: AVAIL: 44   READ: 0   TIME_READ: 0   TIME_SCAN: 15
  GRP: AVAIL: 3   READ: 0   TIME_READ: 0   TIME_SCAN: 6
```

Figure 3-1: Excerpt of raw log file data from **nn** and **xrn**.

xrn begins the record for each session with a line identifying the record as coming from **xrn** and the time at which the record was entered into the log. The record contains one line beginning with `GRP:` for each newsgroup the user was shown. For each newsgroup, **xrn** recorded: 1) How many articles were available to be read in the group(`AVAIL:`). 2) How many articles were actually read (`READ:`). 3) How much time, in seconds, was spent both reading and selecting articles (`TIME:`). 4) Whether one of the "catch up" buttons was used. Catch up buttons in **xrn** allow a user to ignore an entire set of articles with one action. The use of a catch up button in a group is shown by an asterisk.

nn begins the record for each session with a line identifying the record as coming from **nn** and the time at which the record was entered. The record contains one line beginning with `GRP:` for every newsgroup the user was shown. For each newsgroup, **nn** recorded: 1) How many articles were available to be read in the group (`AVAIL:`). 2) How many articles were actually read (`READ:`). 3) How many seconds were spent reading the selected articles (`TIME_READ:`). 4) How many seconds the user spent scanning the subjects of available articles choosing which articles to read (`TIME_SCAN:`).

Table 3.1: Comparison of the number of newsgroups read per session at PARC with number of newsgroups users subscribe to as reported by Jolicoeur.[4]

number of newsgroups	surveyed users reading this many newsgroups	observed sessions reading this many newsgroups
5 - 10	27%	22%
11 - 14	9%	8%
15 - 20	18%	19 %
21 - 30	17%	17 %
31 - 50	9%	18 %
51 - 100	7%	15 %
above 100	8%	1 %

3.5.2 Trends in the data

Once analyzed, the collected data show the following trends which we believe are relevant to users' ability to find interesting articles in the Net News system:

1. Users read an average of 15 newsgroups each session. If we ignore sessions in which users read only 0 to 4 newsgroups as representing "quicky" sessions, our data on the number of newsgroups users read nicely correlates with data from a net wide survey of Net News reading habits taken by Jolicoeur.[11] This correlation helps to establish that the PARC Net News community is similar to the net wide community.

2. Often users read none of the articles they subscribe to. Figure 3-2 is a histogram of the fraction of available articles in a newsgroup that the users actually read. The graph shows that the vast majority of the time, users enter a newsgroup, view a list of the available articles, and then exit the newsgroup without reading any articles. Presumably the users subscribed to the groups because the users thought the groups would contain useful information. This failure to read any of the articles indicates that either the information content of the group really

[4]PARC news reading sessions involving 0 to 4 newsgroups were not counted as they do not give a good indication of how many news groups a user subscribed to.

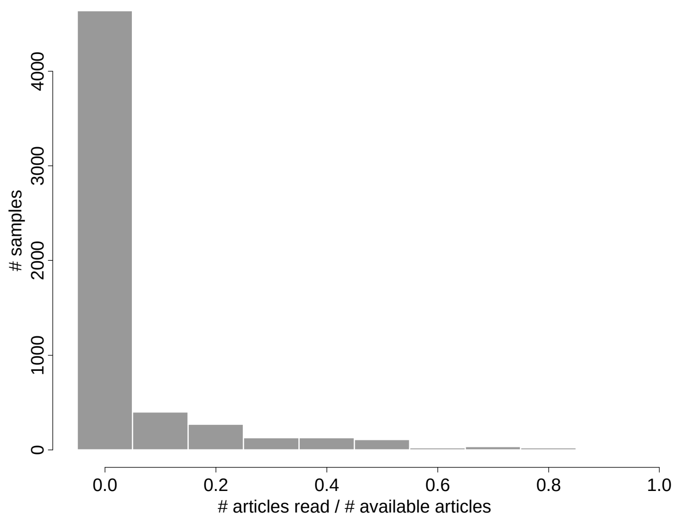

Figure 3-2: Histogram of the fraction of available articles users actually read. The height of each bar represents the number of times users read the listed fraction of available articles.

was very low, or more likely, the user was unable to easily identify any of the articles as being of possible interest.

3. A scatter plot of the number of articles users read versus the number of articles available to be read shows a telling breakpoint at between 200 - 300 articles (see figure 3-3). If there are fewer than 200 articles available to be read in a newsgroup, users read some proportion of the available articles. In groups containing more than 200 available articles however, few users read any articles at all. The common behavior is to simply skip all the articles rather than searching for ones that might be interesting.

4. Far more people read Net News than post. Based on data gathered from 632 sites by the Network Measurement Project at the DEC Network Systems Laboratory, it is clear that for all groups, there are more "lurkers" than "posters."[21][20] This is true regardless of the number of people who read the group or the number who post articles to it. Table 3.2 shows some sample data

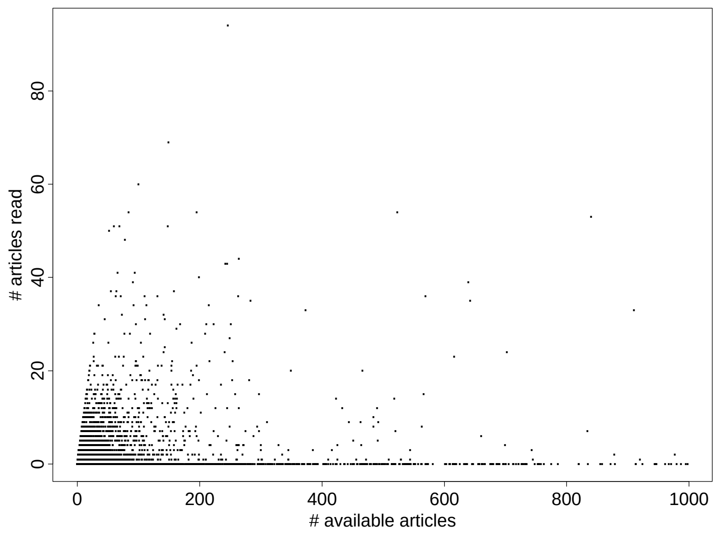

Figure 3-3: Scatterplot of the number of articles available to be read in a newsgroup versus the number of articles users actually read.

for groups with the largest readership, the smallest readership, the greatest traffic, and several others.

5. The time a user spends on each article, even after he or she has taken an action to display the full text of the article, is very short. The data taken so far show that users spend an average of 40 seconds reading an article.

We reduce our observations into a list of design requirements for the collaborative filtering system as follows:

- **Majority Focus** Help the majority of Usenet users. There is a huge number of silent readers — all groups have far more lurkers than active posters. These lurkers are the people we believe collaborative filtering can help the most, and also the people who least want to wade through all the traffic on Usenet. These are also the people least willing to announce their presence (ie: they do not post), and most likely the ones who will least tolerate any extra overhead in their news reading.

Table 3.2: Posting volume and estimated worldwide readership for several news groups. (Data from Reid, USENET Readership report for Aug 93)

newsgroup	est. readers	# posts/month	comment
misc.jobs.offered	200,000	2,016	most readers
rec.humor	150,000	2,087	very popular
comp.unix.questions	130,000	1,062	
talk.abortion	50,000	1,590	controversial
rec.arts.tv.soaps	45,000	5,125	most traffic
bit.listserv.hellas	12,000	802	
aus.sport	2,500	238	fewest readers

- **Streamline** Do not interrupt the existing flow. On average, users spend so little time reading most articles that any operation we expect a majority of users to perform must be exceedingly quick and consistent with the flow of the interface they use for reading articles. Voting for articles must be seamlessly integrated into the way the user processes Net News.

- **Intelligible** The filter must behave in a simple way that is easy to understand. If users can not easily understand how the system is trying to help them, the system will only be getting in the way of the user's tasks.

To help users find interesting articles we need to find a way of cutting down the number of articles they must consider reading, otherwise they will tend not to read any. If we can accomplish that, we will probably help users meet their goal of increasing the number of newsgroups they can read. Our data suggest that even if our filtering is not very accurate at picking out the best articles, it may still help users find articles interesting to them by reducing the psychological burden of sifting through a huge number of available articles.

Where are we? At this point we have established the constraints placed on the design for a Net News collaborative filtering system. The system must provide some method of participating anonymously. The system must not impose excessive overhead on the network. The system may not violate the social conventions of Net News by altering or suppressing messages. It must be easy to install the collaborative filtering system, and the system must not cause irreparable damage in the event of a bug. Finally, the system should help users process articles in a quantifiably faster manner. In the next chapter, we will describe the design of a system which meets these requirements.

Chapter 4

Design Architecture

Since our goal was to develop a robust, scalable system that would interoperate well with Net News, it is not surprising that our system design parallels that of the Net News system in many ways. The fundamental object on which the system operates is called a vote. Each vote represents a single user's evaluation of a single article. The architecture for processing votes is shown in figure 4-1 and it is broken up into two main pieces.

The first piece is the interface module. This interface stands between the news reader programs and the collaborative system. The interface sends votes off into the collaborative system and answers requests from the news reader for information about the popularity of an article. The interface obtains information about the popularity of an article by consulting various vote sources which hold collections of previous users' votes.

The second piece of the architecture is the vote server. The vote server acts as a vote source and is responsible for maintaining a vote database which contains the aggregated votes of all Usenet. The vote server incorporates votes from its own site and other sites into its database, and exchanges the votes cast by local users with the vote servers at other sites. The remaining sections of this chapter will describe each of these parts in more detail, and conclude by comparing the architectural goals we set out previously with the design described here.

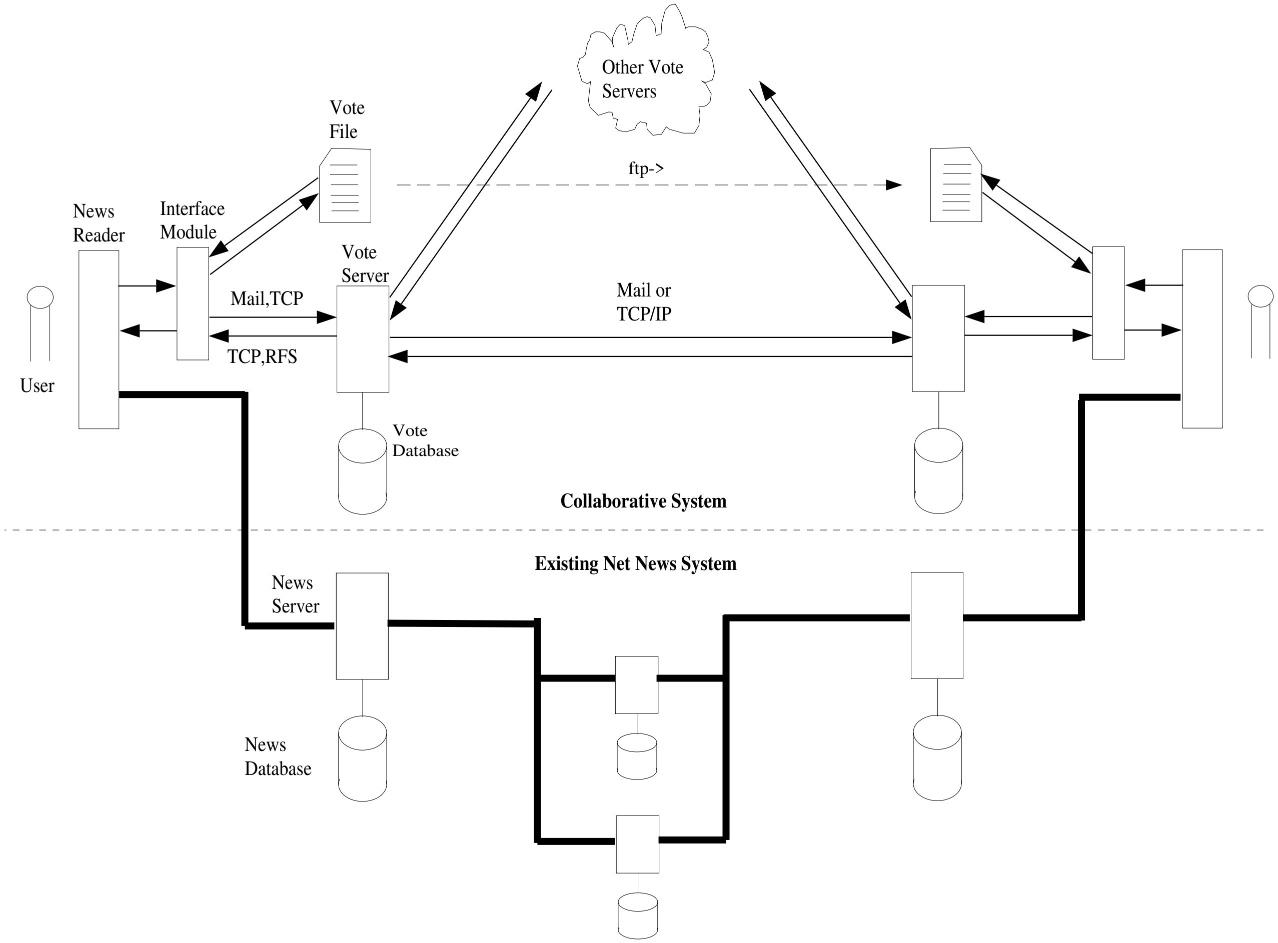

Figure 4-1: Block diagram of the collaborative filtering architecture.

4.1 Votes

Users evaluate the articles they read to create objects we call votes. A vote contains the name of the newsgroup in which the evaluated article resides, the unique message-id assigned to the article, and the user's evaluation of the article. Evaluations are one of *terrible, ok, good,* or *great.*

Although a single article can appear in more than one newsgroup, each vote only contains the name of the newsgroup in which the user read and voted for the article. This decision should make the accumulated votes more useful for filtering as newsgroups are organized around topics. An article cross posted to several newsgroups may be extremely relevant and important in one newsgroup, but totally irrelevant in the others. This decision will also discourage users from posting their articles to more than one newsgroup in hopes of garnering more votes for the article.

4.1.1 Type of information exchanged by vote servers

This thesis did not directly address the issue of what information will be most useful for collaborative filtering. As will be discussed in the last chapter, it is unclear whether numerical rankings from users provide enough information to help other users filter articles. It is also likely that there is other information we could gather inoffensively which would be cheap to transport and useful for filtering. The goal of the vote transport system we created was to be flexible enough to encompass extra information as an add on. Once some system is operational, it will be possible to experiment with exchanging different types of information.

4.2 Vote sources

A collection of votes is termed a vote source. We support two types of vote sources in order to support the two major classes of queries we expect users to make of the collaborative system. The first type of vote source is a specially formatted vote file. The second type of source is a vote server program.

Vote files provide our system with identified votes. Much work has already been done to create filesystems that allow tight control over who can read or write a file. Votes written into a file can be guaranteed both to originate only from one of the people with write permission on the file, and to be read only by those with read permission on the file. The intention is that if an individual creates a vote file, other users can consult that vote source to determine how the individual evaluated an article.

The presence of these individualized vote sources makes it possible to support group and custom moderation as users can create filters of the form, "Show me the articles that Jane Doe liked." The system can answer the query by consulting Jane Doe's vote file provided that Jane Doe has provided the user making the query with access to her vote file. Jane Doe can control which people have access to her opinions in the same way she currently controls who has access to her other files, and could even store her votes for articles in different groups into different vote files to refine further her control over others' access to her opinions.

Vote servers as a vote source provide access to the summarized evaluations of the entire Usenet. When an evaluation of an article is requested from a vote server, the returned response represents a summary of all the votes posted for that article that have been received by the vote server. The information contained in the vote server supports the exploratory searching described earlier as presumably the best articles in a newsgroup will have received the most votes. New users exploring a newsgroup can quickly locate the articles previous users of the newsgroup thought were most relevant. Based on these articles, the users can decide whether or not to continue searching the newsgroup for information.

4.2.1 Naming vote sources

In order for the collaborative filtering system to work with multiple vote sources, there must be a naming convention to describe and identify each vote source. Uniform Resource Locators (URLs) [2] provide such a mechanism so vote sources are referred to by their URL. URLs are especially convenient for vote sources as they contain both the address of the information and an access method. For example, a vote file with a local name of /usr/maltz/votes available by anonymous ftp from intrepid.xerox.com would have the URL *ftp://intrepid.xerox.com/usr/maltz/votes*. A vote server that listened for TCP connections at port 3636 of host news.xerox.com would be known as *telnet://news.xerox.com:3636*.

While sites are assumed to have a local vote server whose URL is well known at that site, some method must exist for informing users of the URL for a vote file. Any of several approaches could be used for informing other users where a person's vote file lives. We think of the location of a person's vote file as just another piece of localizing information similar to a telephone number.

The simplest mechanism for finding a person's vote file would be to ask them personally. This mechanism is identical to the way phone numbers and email addresses are frequently exchanged. A more complicated mechanism would be for users to include the URL for their vote file as an extra header line in their posts to Net News — many people already distribute their phone numbers and addresses in this fashion. News reader programs could take advantage of the extra header line by extracting the URL from articles the user liked. These URLs might then be added to the list of sources the user uses to filter Net News. Another option might be the creation of new newsgroups where users exchange URLs and perhaps discuss which vote files and URLs are the best for filtering various newsgroups.

4.3 Interface module

The interface module provides simple methods for a news reader program to send votes into the collaborative system, or to request information from the collaborative system.

Exactly how a user indicates his or her evaluation of an article is a user interface question left up the designer of the news reader program. The news reader designer can choose whatever mechanism fits in best with the existing user interface. The designer might even choose not to require explicit voting on the part of the user, but rather anticipate the user's vote based on observed behavior. If the user saved or printed the article, he probably found the article very useful and the news reader could record a positive vote (perhaps an anonymous one since the user did not explicitly authorize the vote). If the user killed the author or the subject thread after reading an article, it would be safe to assume the user did not like the article.

For casting votes, the interface module supports two basic mechanisms. Votes can be sent anonymously or identified. The system will attempt to keep anonymous votes from being associated publicly with the user who cast them by sending them on to a vote server where they will be aggregated with other votes. For identified votes, the interface module attempts to enter the votes into the user's current vote file.

Votes cast anonymously are passed directly on to a vote server for inclusion in net wide tallies. Identified votes are sent on the vote server for inclusion in aggregate summaries, but they are also placed in a vote file specified to the interface module. The interface module will also check the validity of the other votes in the vote file at this time (see maintenance mechanisms below).

When answering requests for information, the interface module can combine the votes from several sources, each source being a vote server or a vote file. Combining different vote sources in the interface module makes possible queries of the form: "Show me all the articles that Joe User or Jane User liked." The collaborative information returned by the interface module will consist of lists of articles and the combined evaluations for those articles. The news reader program can use this information as it sees fit to help the user filter his or her news. For example, the news reader could display the information directly to the user, or it could change the presentation order of the articles to show the most recent ones first, or it could use the information as an input to another filtering mechanism such as keyword spotting.

4.3.1 Specification of the Application Program Interface

This section defines the application program interface (API) exported by the interface module. The API provides a structured way for the news reader program to communicate with the interface module and consists of two main parts. The first part deals with how the news reader casts votes for articles, and the second part deals with how the news reader gains access to the collected information for an article. A specification for the procedures in the API is given in table 4.1. The structure which represents a vote is described in table 4.2.

Casting a vote requires only that the news reader provide the interface with the information required to create the vote: the name of the newsgroup in which the article appears, the message-id contained in the article, and the user's evaluation of the article. Before exiting, the news reader must call the procedure *collab_send_off_votes()* to actually send the votes to the collaborative system. If the news reader provides a file name as an argument to *collab_send_off_votes()*, any votes not cast anonymously will be written to that file. The effective uid of the news reader program will need to have read/write privileges on the file.

Before retrieving vote information from the collaborative system, the news reader client must first set up the list of sources (eg: vote files and/or a vote server) that should be searched for votes. The *collab_add_source()* and *collab_remove_source()* routines are provided for this purpose. The strings passed in as source names should be the URLs of the vote files or vote servers. The format of these URLs was discussed earlier.

Once a list of sources has been set, a string containing the names of all the newsgroups which have votes in them can be obtained by calling *collab_available_groups()*. The aggregated votes for the articles in a newsgroup can be obtained by calling *collab_get_votes_for_group()*. This will return a linked list of *vote_t* structures (see table 4.2) each of which contains the vote information for an article. The vote information tells the number of *terrible*, *ok*, *good*, and *great* votes the article received. The linked list of *vote_t* structures should be destroyed when it is no longer needed by calling *collab_destroy_votes()*. Note that *collab_destroy_votes()* does not actually remove the votes from the vote source they were drawn from — this process will be discussed later in section 4.6.

Table 4.1: Specification of the API to the interface module.

```
/******* Routines for casting votes *********/
/* return status: 0 = failed, 1 = successful */
int collab_enter_vote_anonymously(char *message_id, int vote,
  char *group_name);
int collab_enter_vote(char *message_id, int vote,
      char *group_name);

int collab_send_off_votes(char *file_name);
/* send off votes to local vote server and incorporate any
    identified votes into the vote file file_name
   empties the queue of votes waiting to be sent out */

/******* Routines for querying sources for votes  *******/
char* collab_available_groups(int *num_groups);
/* return a new string of the group names separated by \n
   characters;  the number of groups will be left in the loc
   pointed to by num_groups */

vote_t* collab_get_votes_for_group(char *group_name);
/* return a linked list of the votes for articles in group_name
    return NULL if there are no votes available */

/******* change which sources we collect data from *******/
/* return status: 0 = failed, 1 = successful */
int collab_add_source(char *source_url);
int collab_remove_source(char *source_url);

void collab_destroy_votes(vote_t *votes);
/* free the memory for the list of votes returned by
   collab_get_votes_for_group */

int issue_error(varargs : fmt, args);
/* notify someone of problems in the collaborative system */
```

Table 4.2: Structure of vote_t record.

```
/****** Structure of a vote ******/
struct _vote_data_t {
  int  num_high;
  int num_med;
  int num_low;
  int num_terrible;
  char  *stuff;
}; typedef struct _vote_data_t vote_data_t;

struct _vote_t {
  struct _vote_t *next_vote;
  vote_data_t *vote_data;
  char *message_id;
  void *internal_magic;
}; typedef struct _vote_t vote_t;
```

4.4 Vote server

The vote server is the part of the system responsible for accumulating, marshaling and redistributing votes from one site to another. Votes sent to the server by an interface module are accumulated into the server's vote database and marshaled into a packet with other votes to be sent on to neighboring vote servers. Figure 4-2 shows a block diagram of the vote server's construction.

A vote server consists of two main parts. A daemon process called the **voted** and a server process called **vs**. The **voted** accepts connections from interface modules and exchanges information with them. The protocol used in this exchange is documented in Appendix A. The server process **vs** is responsible for maintaining the vote database — adding new votes to it and removing old votes from it — as well as exchanging vote information with other vote servers. The design of the vote server was heavily influenced by the design of the INN news server by Rich Salz [25] and the reference **nntpd** implementation by Phil Lapsley and Stan Barber[15]. The **voted** program reuses substantial portions of code from **nntpd**.

The three files associated with the vote server are the *grouplist, localvotes,* and

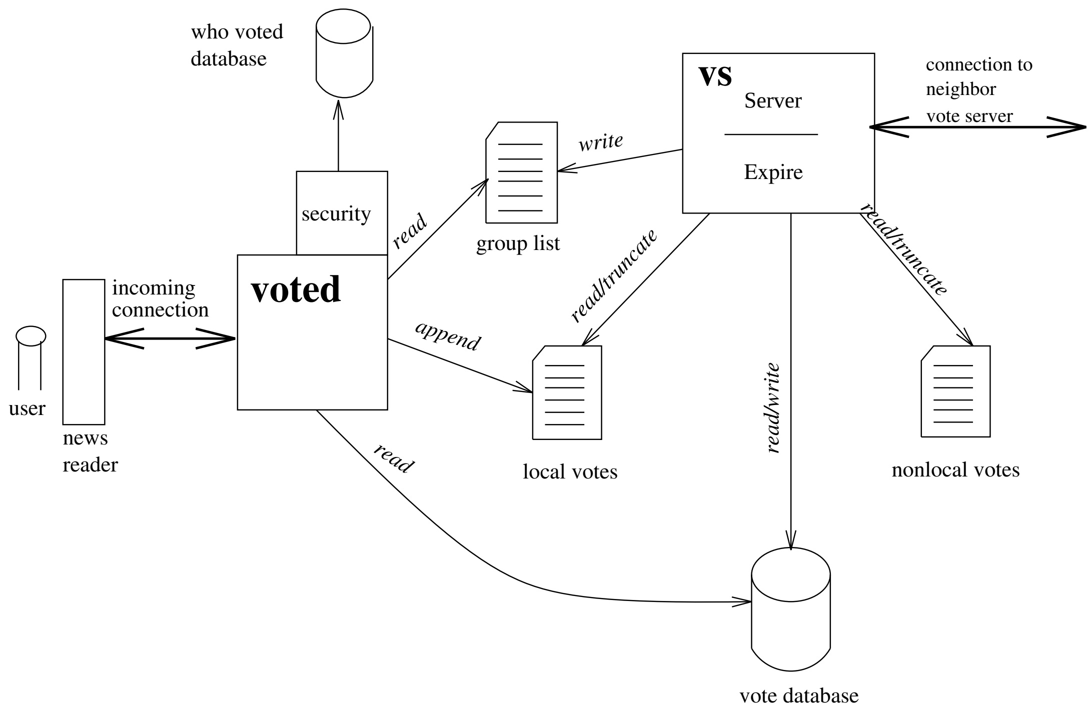

Figure 4-2: Block diagram for vote server. The major components are the daemon (**voted**) which accepts and handles connections from clients and the vote server (**vs**) which updates and maintains the vote database.

nonlocalvotes files. The *grouplist* file is maintained by the **vs** server process and contains a list of all the newsgroups which have votes cast for articles in them. It is used by the **voted** to answer AVAIL requests. The *localvotes* file is created and added to by the **voted**. It contains a list of all the votes cast by users at the local site that have not yet been processed by the **vs**. The **vs** reads the *localvotes* file and incorporates the votes into the local vote database as well as marshaling the votes together into a packet for transmission to other sites. The *nonlocalvotes* file provides a means of entering into the system votes that arrive by mail or ftp from other sites.

The idea of the vote server is to have one running at every site that participates in collaborative filtering. The program would ideally run on the same machine as the news server itself, thereby reducing the number of failure points in the system. This is not a firm requirement however, and vote servers can be added or removed as the load demands. As will be discussed later, vote servers can keep track of which users have already voted on a given article to prevent these users from voting more than once on the same article.

4.4.1 Accumulating votes into summaries

Since we have chosen to handle only numerical votes in this thesis, the vote servers can perform accumulation by simply adding the new votes into running tallies of the number of *terrible, ok, good,* and *great* votes an article has already received. If votes contained more diverse information (perhaps free text keywords) the accumulation process could be easily extended to include a process such as string concatenation. While this would potentially increase the space requirements for storing the summary of collaborative information on an article, techniques such as bounding the size allowed for free text annotations could be used to hold the space requirements down. In the bounded size technique, vote servers would incorporate free text annotations into the vote database until the article had exhausted its allocation of space with any future annotations being silently discarded.

To identify the article to which the vote corresponds, we record in the vote database the full text of the article's message-id. This id is unique to the one article, unlike the article numbers used by the news servers which will vary from server to server. Because the vote servers store their information indexed by a message-id which is unique net wide, users can switch which vote server they use at any time without difficulty. This is a major advantage over the news system itself which requires users to deal primarily with only one news server, since the user's data files are keyed to the article numbers used by that one news server.

We decided to store the message-id as full text rather than as a hashed integer for the sake of compatibility with other systems. The benefit of only storing hashed message-ids is that much less space is required. Some small experiments we did with hashing message-ids shows they can be hashed very effectively into numbers as small as 32 bits by trivial algorithms. We choose to store message-ids in full text form based on the observation that many other Net News utility programs do so, and that to do otherwise would hinder interoperability with these systems. Examples of such

Net News utilities are various news indexing and archiving programs, and **nov** or the "news overview" package[6] by Geoff Collyer, a program which stores information excerpted from article headers in full text form.

4.4.2 Interface module communication with vote server

Numerous options are available for enabling communication between the interface module and the vote server. An early implementation of our system had the interface module mail votes to the server and read votes from the server by accessing a data file exported by the server using the Andrew File System (AFS). To gain robustness and generality, we have followed the lead of the Net News system and chosen Berkeley sockets to be the primary means of communication between the IM and the vote server.

IMs needing to access collaborative filtering information from a vote server open a TCP connection to a well known port where the **voted** is listening. The connection is essentially stateless as all commands are one line terminated by a newline ('\n') and no state is kept by the server between commands. The protocol allows clients to request the names of all the groups that currently contain articles which have votes cast for them; to cast votes for articles, or to retrieve the votes cast for the articles in a group. The protocol for communication is described in detail in Appendix A.

4.4.3 How vote servers exchange information

The vote servers exchange information in a fashion similar to the news servers. At regular intervals each vote server contacts its neighbors and offers to exchange packets of votes with them. Before contacting other vote servers, a server combines all the votes received from local users since the last information exchange into a single packet. The vote server then offers to send each neighbor the packet of new votes and any packets the server has recently received from other neighbors.

When a server receives a packet for the first time, the server combines the votes from the packet into the server's vote database. The server records the packet as having been accepted so the server will not accept the same packet twice. The server continues to store the packet however, so that the packet can be offered to the server's other neighbors. Once the server has offered the packet to all its neighbors, the packet will be deleted.

As a trivial implementation of this, the news system itself can be used to exchange the votes. Consider two sites which wish to exchange vote information. If these two sites do not already exchange Net News, a simple change to their *newsfeeds* or *sys* file can set up the news servers so that they exchange only one group, a new group that will be created just for exchanging ratings. Another one line addition to the *newsfeeds* or *sys* file can cause the news server to hand off each article received in the votes group to a batcher script which prepares the vote packets for input into the vote server. Since the vote packets are processed and fed into the vote server as soon as they are received, the expiration time on this votes group can be set to the shortest possible time. Unlike the case described in section 3.2, the votes group will be minimally small as we are only co-opting the Net News system's transfer mechanism, not its storage mechanism.

If a further reduction in network traffic is required, the system could again take advantage of the summarized nature of the exchanged votes and create a hierarchy of vote servers. In this arrangement, each vote server only sends votes to its parent node and the other children of the parent. The parent node would combine the summaries of its children to form a summary which represented all the nodes under the parent. This super-summary could then be transported up the tree in a repeat of the process.

4.5 Security and privacy of the system

The primary security issue the collaborative system must protect against is ballot-box stuffing. If a large enough group of people start using the system to filter their incoming news, stuffing the ballot-box for or against articles will become a way of harassing and censoring groups of users on the net. To prevent the opinions of any individual from having undue effect on the outcome of the collaborative filtering process, the collaborative system should ensure that each individual can only cast one vote for each article.

Unfortunately, there is a trade off between the desire to allow users to vote anonymously and the desire to prevent ballot-box stuffing. If every vote is transported through the system authenticated with a digital signature[24] from the user casting the vote, it is possible to prevent ballot-box stuffing completely as the final consumer of the votes can ensure that each user votes only once. Gone, however, is the ability to aggregate votes into summaries as each vote must be transported in its digitally signed form. Further missing from such a system is the ability to cast votes anonymously. If preventing ballot box stuffing became crucial to the success of the collaborative system, a more complicated system proposed by Chaum for use with anonymous digital certifications could be used.[4] This certification system involving two additional independent agencies would allow users to vote anonymously while still preserving the end user's ability to verify that each user voted only once. The overhead involved in such a system may well prevent its application to Net News however.

For the purposes of this thesis, we propose a system which is not perfect, but provides protection against the most common means of attack. If users have their votes posted to vote files, then the existing systems for file protection can insure that only authorized users read or write votes in the vote file. Because vote files are associated with individual people, it is not possible to stuff the ballot box using a vote file. Interface modules reading the vote file can insure that the file only votes for each article once.

Preventing a user from sending multiple votes for a single article to a vote server is harder problem to solve. Since the votes distributed by the vote servers are anonymous, there is no way to detect if single user has voted multiple times once the votes are aggregated into the stream of distributed votes. However, some security *can* be provided if the communication between an interface module and a vote server is not anonymous. An honest vote server can enforce a one vote per article per user rule by requiring users to authenticate themselves to the vote server before transmitting any votes. The local vote server can record which users have cast votes for each article, and prohibit the same user from casting a second vote for the article. Under this system the local vote server will know which articles a user has voted on. Since the vote server is local to the user's site, it can be trusted to obey the local policy determining the privacy of personal information in electronic form and presumably maintain the user's anonymity. Many sites now have policies which declare who on the system can legitimately access system logs containing personal information.[5]

Users will authenticate with their vote server by sending their user identification number (uid) along with their votes to the vote server. This will prevent casual attempts at ballot-box stuffing, but would not stop a person who writes a program which masquerades as an interface module to the vote server. This stuffing program could generate an arbitrary sequence of numbers for uids — effectively casting votes for all the users on the system. Another approach for securing the system would be to require the transmission of an actual password as part of the vote server protocol. Such an authentication mechanism has been proposed as part of NNTP II, the second version of the Net News Transfer Protocol, but appears unlikely to be implemented.

Vote servers can be configured to exchange votes only with specific peers, so there is little risk of an individual inserting fake vote packets into the stream of vote packets exchanged between honest vote servers. Unfortunately, there is little that can be done to stop dishonest vote servers from generating fake vote packets to stuff the ballot-box. Once a dishonest vote server is identified (most likely by human operators), the only option to secure the system is for peers to derecognize the dishonest vote server and cease exchanging votes with it. This approach is currently used on Net News itself to control users who forge messages.

4.6 Maintenance mechanisms

An essential component of the Net News system is the expiration of old articles. Since news articles are created at such a tremendous rate, few if any sites can afford to store more than a small fraction of them for any length of time. The Net News system works around this by expiring or removing all articles older than a certain age as part of a routine task that executes every night. Consequently, vote sources will contain vote information for articles that are no longer available unless steps are taken to expire the vote information as well.

There are two reasonable means for expiring vote information. The first option is to check each vote against some news server, presumably the local one. If the article to which the vote refers is still available from the news server, the vote information about the article is retained, otherwise it is discarded. The second option is to associate a time stamp with the vote information for each article. The vote information can then be expired after it reaches some age just as old news articles are expired. The current implementation of the system allows only one global expiration age to be set, but a trivial extension to the system would allow different groups to be expired at different rates.

Since the vote server is an independent process, it can incorporate expiration as one of its tasks to perform on the vote database. Vote files are harder to work with since there is no process directly associated with them. For this reason, whenever new votes are added to a vote file by the collaborative interface module, the IM arranges for the votes already in the vote file to be examined for validity.

4.7 Comparison of system with design criteria

This architecture meets the goals set out in the design criteria section. The vote servers keep the vote information in a separate stream from the news articles so that system administrators do not have to modify their current news servers. By placing a vote server local to every participating site and providing a means for them to exchange information between themselves, there is no one central vote server authority. Since votes are stored separately from the articles, there is no change made to any article. Any user who wants to see all the articles posted to a newsgroup in their original form can do so by simply ignoring the votes, so there is no issue of censorship. The API provides an interface between the collaborative system and the news reader program, so modifying existing news readers to use collaborative filtering should be easy (this is discussed further in chapter 5). Integrating collaborative filtering into existing news reader programs will both prevent users from having to learn a new user interface, and allow news readers to use the collaborative filtering data in whatever way the program's user interface supports best. Users can vote in either an identified or anonymous fashion, and control who has access to their opinions. Most importantly, the space and bandwidth requirements for the system are very low due to the summary nature of much of the data.

In the next chapter we will present data from the use of the system, notes on our implementation of the client news readers, and finally conclusions we have drawn from this project.

Chapter 5

Results and Conclusions

In this chapter we will first describe our implementation of the system design presented in the previous chapter. We will then show some initial results from the use of the system, and finally draw conclusions from our experience designing, implementing and using the collaborative filtering system.

5.1 Implemented system

The implementation of our collaborative filtering system proceeded in a see-saw fashion: development efforts first focused on the news reader clients, then the system's backend (the interface module and vote server), and then shifted to focus on the news reader clients again. In the subsections that follow, we first give a chronology for the development of the system. We then describe in detail the way the system appears to its users through the user interfaces of the two news reader programs we modified to support collaborative filtering.

5.1.1 Time line for development

The first implementation of the system was a quick hack to create a working prototype that could be used to better define the problems that needed to be solve. Later implementations made the system cleaner and brought it in line with what came to be the final design described in the previous chapter. Our implementations can be roughly grouped into four phases. The first phase involved modifying the news readers to collect statistics on user behaviors. The second phase was creating a simple vote server and hacking into the news readers a method for user voting. In this phase, votes were mailed to the vote server from the news reader. The third phase hacked the news readers to read vote summaries out of a file and then use the summaries to filter or sort articles for the user. The forth phase involved writing the interface module(IM) and the expiration mechanisms; pulling the hacked vote communication out of the readers; and making the news readers use the IM for communicating votes with the rest of the collaborative filtering system. The fourth phase also cleaned up the communication between news reader, IM, and vote server.

The development of the vote server and interface module was fairly straightforward and was accomplished without any significant or interesting difficulties. Since the goal of creating the interface module was to make it easy for programmers to add collaborative filtering to existing news readers, it was very satisfying to find it trivially easy to modify the **nn** and **xrn** news readers to traffic in collaborative filtering information. The only difficulties came in modifying the news readers so an article's message-id was available to the filtering routines when we wished to decide whether or not to display the article to the user. The filtering routines needed access to the articles' message-id in order to find the articles' vote histories since the vote information is indexed by message-id. The difficulties arose because the internal data structures of **nn** and **xrn** were not well set up to handle article message-ids.

5.1.2 The news reader clients

The most visible parts of the system we implemented are the news reader clients used by users to read their news. So long as it is working properly, the remainder of the collaborative filtering system sits out of sight of the end users. We had originally chosen the **xrn** and **nn** news readers to gather usage statistics from because they are both popular news readers. Together, they also cover a range of interfaces from the text-based interface used by **nn** to the graphical user interface of the X based **xrn**.

For these same reasons as well as our new found familiarity with their source code, we chose to modify **nn** and **xrn** to support collaborative filtering.

In modifying the news reader clients, our primary goal was to make it as easy as possible for users to vote and filter. As we previously explained, users spend so little time looking at individual articles, that we wanted to streamline the voting process into their natural news reading pattern. At the same time however, we wanted to subtly remind users that they could, and should, vote for or against the articles that they liked and didn't like. Finding efficient ways to get votes from users and present filtering information to them was the key challenge of modifying the news readers.

5.1.3 Modifications to the xrn news reader

A picture of the user interface presented by the modified **xrn** is shown in figure 5-1. When users start up the program, two windows appear on their screen: the main window (shown on the right of the figure) and the filtering control panel (shown at the upper left of the figure). The main window appears as it does in the unmodified version of **xrn**. All the controls for the collaborative filtering system appear in the control panel window. This allows users who do not want to bother with the filtering system an easy way to dismiss the filtering controls, while still reminding them that the system is there.[1] Consistent with the rest of the **xrn** interface, as the user moves the cursor over the buttons and toggles of the control panel, they are shown short help messages explaining what the button or toggle does. More detailed information on the collaborative system is available by selecting the "Info" button.

When the user enters a newsgroup to read the articles available in it, the current filtering mode is used to determine how the available articles are presented to the user. In the "normal" filtering mode, articles are presented to the user without any changes by the filtering system. In the "Only popular" filtering mode, only articles that have been judged to be popular by the system are displayed to the user. In the "Popular

[1] An initial release of the modified **xrn** client placed the filtering controls in the main window and resulted in complaints that the new buttons had caused the original buttons to move to different locations. This change resulted in users having to relearn their mouse-clicking patterns and clearly violated our efforts to seamlessly integrate collaborative filtering into the news reader.

Figure 5-1: Interface for the **xrn** news reader client. Main window on right, filtering controls on left.

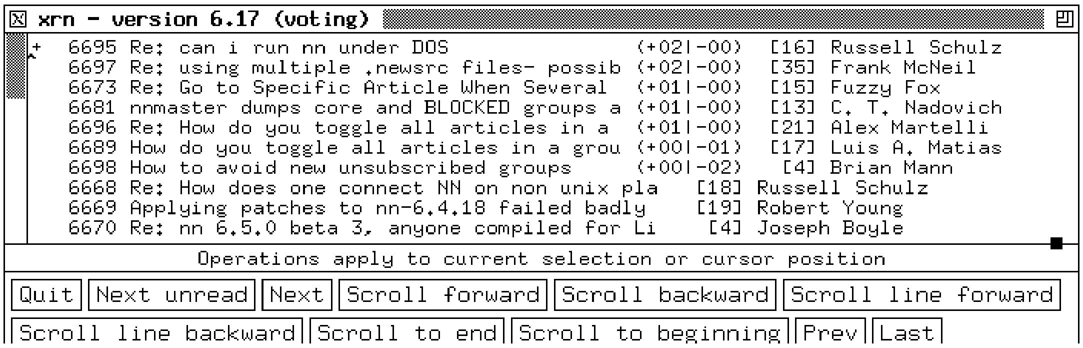

Figure 5-2: Portion of **xrn** main window showing a list of available articles sorted by order of popularity. The numbers in '()' represent the number of positive and negative votes received by the articles.

first" mode, the articles are sorted by order of popularity. Articles receiving the most votes are placed at the top of the list of available articles, and articles receiving no votes are placed at the bottom of the list. Articles receiving mostly negative votes are placed together in a block after the articles which received positive votes, but before the articles that have received no votes. In both the "Popular first" and "Only popular" modes, articles are presented to the user along with numbers indicating how many people have voted for and against the article. These rating numbers are to help the user gauge the importance or relevance of the articles (see figure 5-2).

Users can vote on an article at any time while they are reading it by clicking on the "Great," "Good," or "Bad" buttons. If the user has set the "Auto advance" feature, they will be automatically taken to the next screen or next article when they vote on an article. This means that reading and voting for an article takes no more mouse-clicks than just reading it would — a feature which we believe substantially streamlines the interface.

As people first begin to use the collaborative filtering system, very few of the available newsgroups will contain any articles that have been voted on. To help users find newsgroups which have collaborative filtering information available, we created the 'group list' menu shown on the bottom left of figure 5-1. This menu is opened from the filtering control panel and shows all the groups for which some filtering information is available. If a user clicks on a listed group, the user is immediately taken to that group and shown the available articles in the group. The goal is that by helping users find the groups in which some votes have already been cast, we can help those groups quickly achieve a critical mass of users.

5.1.4 Modifying the nn news reader

For several reasons, **nn** interacts with users in a very different way than **xrn**. The first difference is that being text based, **nn** requires users to memorize key stroke commands rather than presenting them with a graphical display of buttons to press. The second main difference is that **nn** uses a two tier presentation scheme. For each newsgroup to which the user is subscribed, he is first shown a list of the subjects for all the available articles in the newsgroup. At this point, the users selects which articles to read based on the subjects of the articles. Next, the selected articles are shown to the user, with the unselected articles being skipped over.

A picture of the "article selection" tier from the **nn** interface is shown in figure 5-3. In the modified **nn**, the collaborative filter is run whenever a user enters a newsgroup. The most popular articles among the ones displayed are automatically selected by the collaborative filter, so that if the user wants to read only the popular articles she can go directly into the article reading tier. If the user wishes to, she can change the number articles autoselected by the filtering system, or manually select and unselect articles as in the unmodified **nn**. The interface displays at the bottom of the screen how many articles were autoselected by the filtering system, and can also show the user a list of only the selected articles.

Once the user has moved into the second "article reading" tier of the interface, the display changes to look like that of figure 5-4. We attempt to remind the user to vote for articles by placing a line at the bottom of the screen describing the voting keys. **nn** is extremely configurable, and by setting a variable users can select an auto-advance on vote mode similar to the one implemented for **xrn**. With this mode set, users are automatically sent to the next screen of text or the next article after voting so that voting on an article requires no more key strokes than simply reading that article does. More information about the collaborative system is available to the user from the help menus, and from the message-of-the-day displayed at **nn**'s startup.

5.2 Results

Our collaborative filtering system was in use at Xerox's Palo Alto Research Center(PARC) from approximately August 1993 till December 1993. During this time, the number of votes cast by users of the system were recorded, and are depicted in figure 5-5. Each bar represents the number of votes cast during a week long period beginning on a Sunday. Over the period of this study, 44 users voted at least once. 24 of the voters used the **nn** news reader and the other 20 voters were **xrn** users. There

```
Newsgroup: rec.climbing                                     Articles: 28 of 45/8

a Christian Lowe    74  >Ice Climbing Gloves
b Arthur Kingston   52  Illogical reverence
c Eric Coomer       16  >
d Douglas Meredith  87  Forest Service Reinvention Meeting -- Phoenix
e Bob Breivogel     30  >>>>>>Sigg bottle ??
f Steven Reiser     21  >>>>>>Illogical reverence of fi<>enters (was Trip Report)
g Steven Reiser     51  >>>>>>
h Steven Reiser     15  Preclipped Leading
i Steven Reiser     63  >>Bolts, Bolts, Bolts, Bolts, Bolts, Bolts...
j wayne trzyna      30  >
k JOHN EVANS         1  Climbing in Hong Kong?
l Steven Reiser     50  >Get that "great runout fe<>" without the bad conscience!
m schneider@eng     37  -
n MarcJ 512          9  >Life of a bolt?????????????
o YVHS              12  >Small Cams
p Jerome Borrel     18  Climbing in France ?
q Quang-Tuan Luong  31  >
r$Eric Toler        10  >>Climbing area TRASHED!
s Scott Linn        32  >Beacon Rock - bolts chopped

-- 19:41 -- SELECT -- help:? -----Top 66%-----
there were 3 messages auto selected by the collaborative system
```

Figure 5-3: The first tier in the **nn** interface: selecting the articles to read. The highlighted article marked with a '$' has been autoselected by the collaborative filtering system.

```
Garry N. Ray: >>>>Prussik Sling on Rappel : Rope Size?  Wed, 27 Apr 1994 18:21
The riotous muse compelled Jefferson Svengsouk (Svengsouk.1@osu.edu) to exclaim:

: upon safety factor, and probably makes allowances for rescue and other high
: load uses, 5 mm should be the max. But as you and I have experienced, in
: most uses, 7 mm doesn't seem to present a problem.

5 to 7 mm works fine. I've used sewn runners in lieu of perlon. They grab
just as quickly if you run them in say 5 or 6 loops around the rope
(spectra seems marginally less effective here).

I second the notion of having a pre-tied second prussik to get your weight
OFF of the primary sling. Try this at home, under controlled conditions,
to discover the ridiculous helplessness of *not* carrying a second sling
on you (that's where runners come in handy). You will laugh when you
discover just how stuck you can become.

*PRACTICE* this maneuver as per the Texas prussik in Freedom Of The Hills
or elsewhere. You will hate learning this technique at 500 feet, under an
overhang.

-- 19:50 --rec.climbing-- 1 MORE --help:?--Top 62%--
hit #'s 0-3 to vote:0=terrible, 1=good, 2=great, 3=fantastic
```

Figure 5-4: The second tier in the **nn** interface: reading the selected articles. Users are reminded to vote by the line at the bottom.

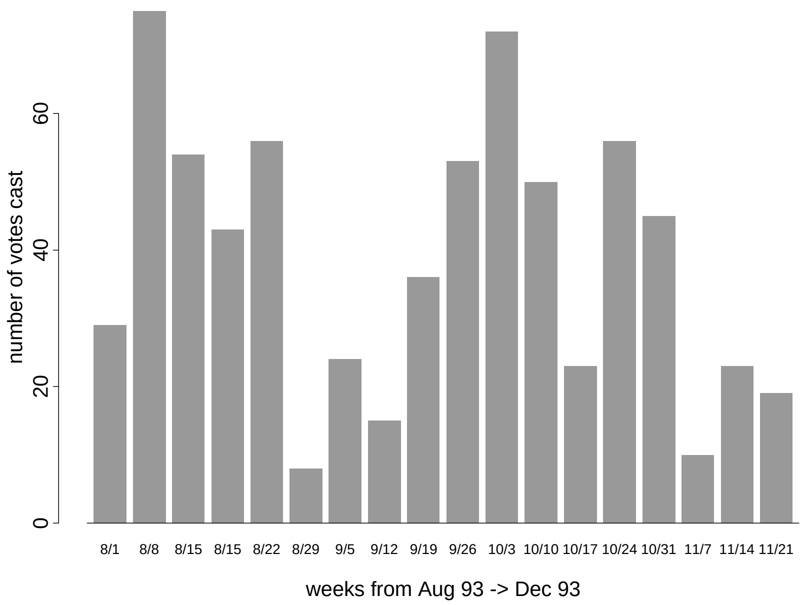

Figure 5-5: Number of votes cast at Xerox PARC during the period from August 1993 till December 1993. Each bar represents the number of votes cast during the week beginning on the listed Sunday.

are approximately 40 users of **nn** at PARC, so roughly half of the **nn** users voted.

Over the course of the study, the fewest votes were cast the week of August 29th. Coming directly before the holiday of Labor Day weekend, it is possible that this lull in voting was caused by many Net News users taking vacation time. The failure of the voting level to return to its previous level after the holiday weeks may be attributable to a cause identified by a group at Bellcore researching ephemeral interest groups.[3] This group found that adding features unobtrusively to commonly used programs resulted in users frequently forgetting about the new features. One possible explanation for the dip in our voting levels was that people went away on vacation, and then completely forgot about the voting options when they returned to work. Perhaps our reminders to vote were too subtle.

The upswing in voting the week of 9/26 coincides with the release of a new version of the news readers and the sending of an email message to a site-wide mailing list reminding people of the system's existence and how to vote. After this resurgence, the voting levels appear to again drop slowly over time.

Examination the voting records indicates that the collaborative filtering system did not achieve anywhere near a critical mass of users. In almost every newsgroup in which any votes were cast, all the votes came from a single user. Had several users contributed votes on the same article, we would have taken this as a sign that users were paying extra attention to articles their peers had read and voted on. In examining the effect of the votes which were cast, it appears that there is so little overlap in which newsgroups are read by the users of **nn** and **xrn** at PARC that it is likely few users had articles pointed out them by the collaborative filtering system.

5.3 Conclusions

The collaborative system we created did not receive anywhere near enough use to be able to determine whether or not the type of collaborative filtering implemented by the system is effective at helping users find interesting articles. Even given that limitation, however, there are still conclusions we can draw from our experience designing and implementing a collaborative filtering system for Usenet Net News.

Our primary conclusion is that it is both possible and feasible to create a system that will efficiently transport some types of vote information from the users who create it to users who will use it to filter their Net News. Our design and implementation represent an existence proof of such a system that provides robustness, ease of integration with existing Net News clients and servers, and low overhead on the communications network. Relating to this primary conclusion are a number of secondary ones that follow from it.

Our first supporting conclusion is that there exist some groups of users, specifically the ones at sites like PARC, who are willing to take some amount of time and explicitly vote for and against the Net News articles they read. Although there was no tangible reward offered for doing so, over half the users of the **nn** news reader at PARC cast votes. This bodes well for the future success of collaborative filtering systems as it indicates that for whatever reason, users are willing to contribute their expertise to a system which advances a global good.

Our second supporting conclusions relates to the specific domain of Net News. Since a collaborative system necessarily involves transporting information from one user to another, meta-issues such as privacy and social conventions must be addressed by the system. This can be particularly challenging on Net News where many users exhibit strong convictions over what is and is not acceptable practice. Our resolution to these problems was to avoid making changes to the Net News system itself or the data it contains. To make users feel comfortable with disclosing information about which news articles they read, we provide a method of anonymous voting that still contributes significantly to the effectiveness of the collaborative filtering system.

Our third and final supporting conclusion was finding that as in many groupware systems, critical mass is the show stopper. Until the system is well in use, there are uncertain rewards for any user who participates. With a system such as ours, we can define critical mass as having been reached in a newsgroup when users derive benefit from the votes cast by other users, and in turn cast votes themselves. Our data indicates that our system at PARC never reached critical mass in any newsgroup.

5.4 Future work

Working with our collaborative filtering system system has highlighted several questions that are in need of further effort to resolve. The key undecided issue is whether or not a system of the type we designed will actually help users to filter their Net News. The final answer to this question will most likely require the installation and use of our system in a far larger setting than that just PARC. We anticipate the success or failure of the system will be affected by many things, including the social interactions between eager and lazy readers[13], the minute details of the user interfaces, and the ways in which filtered articles are presented to users. Still other unanswered questions important to success of a collaborative filtering system are given below.

A major issue that is a direct result of our work is deciding what types of information should be collected from users and distributed by the collaborative filtering system. While our initial impression was that having users evaluate articles on a scale from “terrible” through “great” would be the clearest and easiest task for them, several users reported that this is not so. They point out the difficulty of evaluating articles that might be very funny and interesting, but completely factually incorrect — are those “good” articles or “terrible” ones? They suggested that they would find it easier to make a series of binary decisions in a multi-dimensional space than to try to collapse their evaluation to a point on a more finely divided single dimensional scale. Perhaps asking the users to rate articles as being “correct” or “incorrect” and “interesting” or “not interesting” would be a better evaluation method. As noted in section 4.1.1, it would be easy to modify our vote servers and interface module to support such information.

Related to the question of how users evaluate articles is the question, “Do users need to rate articles explicitly at all?” We might assume that users would want to cast explicitly any vote identified with them, but users might not be so worried about the accuracy of a vote cast anonymously. For these anonymous votes, it should be possible to estimate a user’s opinion of an article simply by noting how the user processes the article. An article skimmed quickly and passed over might rate a low positive vote. An article the user saved or printed could receive a high positive vote, and an article that caused the user to purge all articles on the same subject could safely be given a negative vote. Studying the behaviors of users as they manipulate a news reader could provide interesting insight on how to automatically extract user evaluations from user behavior.

Another issue left unresolved by this thesis is the security of votes and vote files. For vote files to be truly useful, there must be better and simpler means of controlling who has the ability to access or create identified votes. Our approach of using ftp and file system permissions, while workable, is clearly not an ideal solution. To a large extent, vote files can be viewed as yet another network information resource, and so work on universal resource names (URNs) may solve this problem.

Finally, there is no reason that our techniques of collaborative filtering could not be applied to other types of distributed information resources. Since our vote system is free standing, it should be relatively straight forward to use our framework to compile ratings on other systems such as Gopher or World Wide Web(WWW). As an example of how our system could interact with WWW, we can imagine indexing votes by URL just as we currently index votes by message-id. We could group together the votes for all the links out of a Web page just as we currently group together the votes for all the articles in a newsgroup. Such a system would not only help users navigate the Web by providing information about the most popular Web pages and the most popular routes through the Web, but would also create within the vote server a map of the Web. If the votes were periodically expired, the map would keep itself up to date as Web pages appeared, disappeared, and were improved over time.

In ending, it seems clear that the rapidly increasing bandwidth and interconnectivity between computers has resulted in users being deluged with more information than they can possibly use. Yet, through techniques such as collaborative filtering that same interconnectivity can help us find the gems in the haystack by putting humans in better contact with other humans.

Appendix A

Vote Server Communication Protocol

This appendix describes the protocol used by the vote server to exchange information with interface module clients. The protocol is essentially stateless as all commands are one line terminated by a newline ('\n') and no state is kept by the server between commands. Bodies of data are terminated by a period ('.') on a line by itself. All commands are case-insensitive, all bodies of data are case-sensitive. Any errors are silently discarded. Message-ids should include the delimiting '< >'. The vote server may break the connection with the client if no messages are sent over the connection within some reasonable length of time.

A.1.1 AVAIL - what groups have votes available

syntax: AVAIL

reply: <number>
<group.name.one>
<group.name.two>
...
.

The vote server returns a newline separated list of all the groups for which the server holds votes. The number preceding the list is an approximate count of the number of group names that will follow. It is used inside the current interface module implementation as a hint on how much memory to allocate. The list of group names is terminated by a period on a line by itself.

A.1.2 GROUP - fetch all votes for articles in group

syntax: GROUP <groupname>

reply: <message-id> <n_{high}> <n_{med}> <n_{low}> <$n_{terrible}$> <stuff>
<message-id> <n_{high}> <n_{med}> <n_{low}> <$n_{terrible}$> <stuff>
...
.

The vote server returns all the available filtering information for the articles in <groupname>. The data for each article appears on a separate line beginning with the message-id for the article, followed by the number of high, medium, low and hate-it votes received by the article (n_{high}, n_{med}, n_{low}, and $n_{terrible}$ respectively). Each line is ended with any text annotations that might be associated with the article. The maximum length of the text segment is set in the vote server conf.h file and is currently 1024 characters. The stuff field is left over from early implementations and its contents are not specified.

A.1.3 VOTES - cast votes for articles

syntax: VOTES
GROUP: <groupname>
<message-id> <n_{high}> <n_{med}> <n_{low}> <$n_{terrible}$> <stuff>
...
GROUP: <groupname>
...
.

reply: none.

The data segment to a vote command consists of a list of groups and votes to be cast for articles in the groups. The data for each group begins with the tag 'group:' followed by a group name. The votes follow, one per line, and consist of the article's message-id and the number of high, medium, low and terrible votes to give to the article. If the sum of these fields is not 1, the vote is discarded. The stuff following the rankings field may contain free text annotations, but this field may be truncated, edited or discarded entirely by the vote server.

A.1.4 QUIT - break connection

syntax: QUIT

reply: none.

Break connection with the vote server.

Bibliography

[1] Anklesaria, F.; McCahill, M. et al.; "The Internet Gopher Protocol," RFC 1436, University of Minnesota, March 1993.

[2] Berners-Lee, Tim; "Uniform Resource Locators," working draft, CERN, July 1993. available via ftp from info.cern.ch as /pub/www/url6.[txt,ps]

[3] Brothers, Laurence et al.; "Supporting Informal Communication via Ephemeral Interest Groups," *Computer-supported cooperative work, Proceedings of the 4th conference, CSCW '92 (Toronto, Canada)* ACM: New York, 1992, pp. 84-90.

[4] Chaum, David; "Security without Identification: Transaction Systems to Make Big Brother Obsolete," *Communications of the ACM,* October 1985, Vol 28, No 10, pp. 1030-1044.

[5] Clement, Andrew Guest Editor; "Special Issue: Workshop and panel of Privacy Considerations, CSCW'92," ACM SIGOIS Bulletin, July 1993 Vol 14 No 1, pp. 3-42, especially pp. 6-7.

[6] Collyer, Geoff; *The News Overview Package (nov),* available as part of the INN news server, see [25].

[7] Danzig, Peter; Obraczka, Katia; Li, Shih-Hao; "Internet Resource Discovery Services," *IEEE Computer,* September 1993, pp. 8-22.

[8] Deutsch, Peter; *Resource Discovery in an Internet Environment,* Masters Thesis: School of Computer Science, McGill University, Montreal, June 1992. available via ftp at archives.cc.mcgill.ca as /pub/peterd/peterd.thesis

[9] Fischer, Gerhard; Stevens, Curt; "Information Access in Complex, Poorly Structured Information Spaces," *Human Factors in Computing Systems, CHI'91 Conference Proceedings (New Orleans, LA),* Addison-Wesley, 1991, pp. 63-70.

[10] Johnson, Mark; "SIMON HOMEPAGE: Welcome to SIMON," University of London, available via World Wide Web at http://web.elec.qmw.ac.uk:12121/

[11] Jolicoeur, Lucie; *Utilisation des méthodes de recherche d´information pour le filtrage des News de Usenet,* Masters Thesis: Département d´informatique et de recherche opérationnelle, Université de Montréal, Montreal, April 1994. Also in "Newsreaders survey RESULTS (long!)", Usenetnews.software.readers, 10 Aug 93, Message-ID:<1993Aug10.205558.18192@IRO.UMontreal.CA>; [185 lines].

[12] Kantor, Brian; Lapsley, Phil; "Network News Transfer Protocol," RFC 977, U.C. San Diego, February 1986.

[13] Goldberg, David; Oki, Brian; Nichols, David; Terry, Douglas B.; "Using Collaborative Filtering to Weave an Information Tapestry," *Communications of the ACM,* December 1992, Vol 35, No 12, pp. 61-70.

[14] Hill, William; Hollan, James; Wrobleski, Dave; McCandless, Tim; "Edit Wear and Read Wear," *Human factors in computing systems: Striking a balance. Proceedings of the 9th annual conference. SIGCHI '92 Proceedings (Monterey, CA),* Addison-Wesley, 1992, pp. 3-9.

[15] Lapsley, Phil; Barber, Stan; "nntpd - The reference implementation of the Net News Transfer Protocol," (often found as part of the C-news news server), available via ftp from ftp.std.com as /src/news/nntp1.5.tar

[16] Loeb, Shoshana; "Architecting Personalized Delivery of Multimedia Information," *Communications of the ACM,* December 1992, Vol 35, No 12, pp. 39-48.

[17] Loeb, Shoshana; Yacobi, Y.; "Private Information Filters in a Public Network," available from author: Bell Communications Research, 445 South Street, Morristown, NJ 07962.

[18] Malone, T. W.; Grant, K.R.; Turback, F.A.; "The Information Lens: An Intelligent system for Information Sharing in Organizations," *Human Factors in Computing Systems, CHI'86 Conference Proceedings (Boston, MA),* ACM: New York, 1986, pp. 1-8.

[19] O'Reilly, Tom; Todino, Grace; *Managing uucp and Usenet,* O'Reilly & Associates, Inc., 1992.

[20] Reid, Brian; "USENET Readership report for Aug 93," Usenet news.lists, 9 Sep 1993, Message-ID: <26m3ae$n5k@usenet.pa.dec.com>; [2342 lines].

[21] Reid, Brian; "USENET READERSHIP SUMMARY REPORT FOR AUG 93," Usenet news.lists, 9 Sep 1993, Message-ID: <26m39n$n4l@usenet.pa.dec.com>; [394 lines].

[22] Reid, Brian; "USENET READERSHIP SUMMARY REPORT FOR MAY 93," Usenet news.lists, 2 Jun 1993, Message-ID: <1uibjb$8de@usenet.pa.dec.com>; [402 lines].

[23] Resnick, P.; Iacovou, N.; Suchak, M.; Bergstrom, P.; Riedl, J; "GroupLens: An Open Architecture for Collaborative Filtering of Netnews," MIT Sloan School WP #3666-94, Center for Coordination Science WP #165. To appear in Proceedings of the 1994 ACM Conference on Computer Supported Cooperative Work.

[24] Rivest, R.; Shamir, A.; Adleman, L.; "A Method for Obtaining Digital Signatures and Public Key Cryptosystems," *Communications of the ACM,* February 1978, Vol 21, No 2, pp. 120-126.

[25] Salz, Rich; *The INN News Server system,* available via ftp from ftp.std.com as /src/news/inn1.4.tar

[26] Sheth, Beerud; Maes, Pattie; "Evolving Agents for Personalized Information Filtering," *Proceedings of the Ninth IEEE Conference on Artificial Intelligence for Applications,* 1993.

[27] Terry, Douglas B.; "A Tour through Tapestry," *Proceedings ACM Conference on Organizational Computing Systems (COOCS),* November 1993, pp 21-30.

[28] Weinstein, Sydney; "Special Issue: Network News," *The C Users Journal,* February 1991, pp. 109-116.